Attaining High Assurance in Composed Systems

Jess Irwin, Northrop Grumman Corporation
Information Architect

Gordon Uchenick, Objective Interface
Senior Mentor / Principal Engineer

W. Mark Vanfleet, National Security Agency
Senior INFOSEC Systems Security Analyst

Agenda

- The Problem
- Layered Assurance
- Distributed Systems require Distributed Trust
  - Security and the Services Oriented Architecture
- The Path Forward

Where We Are Now: Development

- Systems are complex now and will be more complex in the future
  - Increasing reliance on software content
- Complex systems are developed in layers and stages
- Layer developers divide implementation into modules
  - “Divide and conquer”
  - “Reuse, don’t reinvent”
- This paradigm is so well accepted it has become a ubiquitous mantra

Where We Are Now: Assurance

- Certification and accreditation are “whole system” regardless of layered development
  - Higher assurance levels increase the certification rigor
- Derivation of total system accreditation from certification of its components is not defined
  - There is no “Risk Algebra”
- This disconnect leads to increased cost and risk
  - Offsets savings of layered architecture and modular implementation

Why is This?

- Failure of system to be certified / accredited at high assurance is intolerable
  - That is why it is a high assurance system in the first place!
- Two reasons why we always inspect the entire system at high assurance levels
  - Assurance argument may not decompose along architectural lines
  - Unpredictable side effects among components

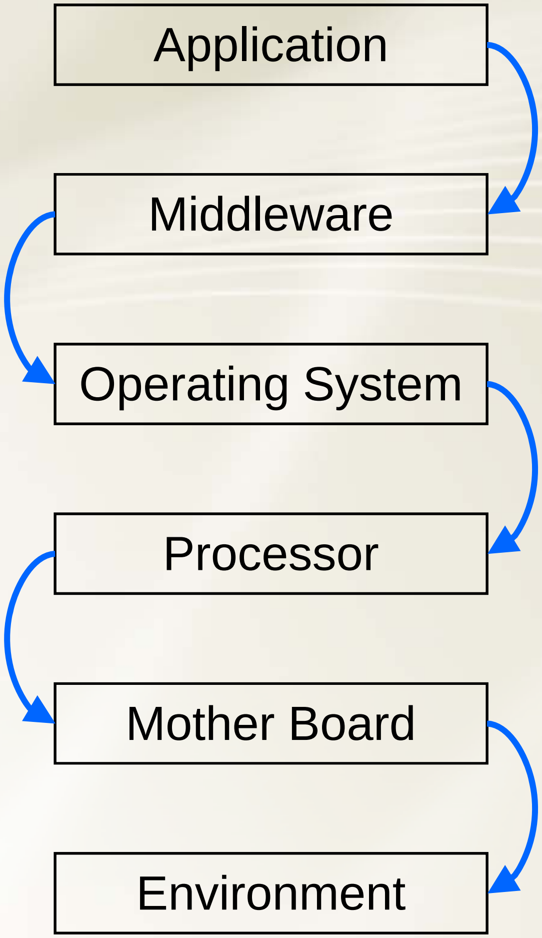
Additive Composition

- Modules retain their properties when integrated
- What we want:
  - Properties (A) || Properties (B) = Properties (B) || Properties (A)
  - Properties (A) || Properties (B) = Properties (A + B)
- True when
  - Properties (A) $\perp$ Properties (B)
  - i.e., when A and B are orthogonal

The Expensive Problem

- We can reuse evidence about a standardized software module
  - Verifying the evidence for a single module is straightforward
- We can’t reuse the effort put into evaluating that the composition of modules has required properties
  - Every system combines standardized modules in different ways
  - Module composition is not additive
    - Properties (A) || Properties (B) ≠ Properties (B) || Properties (A)
    - Properties (A) || Properties (B) ≠ Properties (A + B)
  - Verifying composition of modules is an art, not a science
    - Because of this, we rely heavily on testing

Simplistic Layered Trust



Application

Middleware

Operating System

Processor

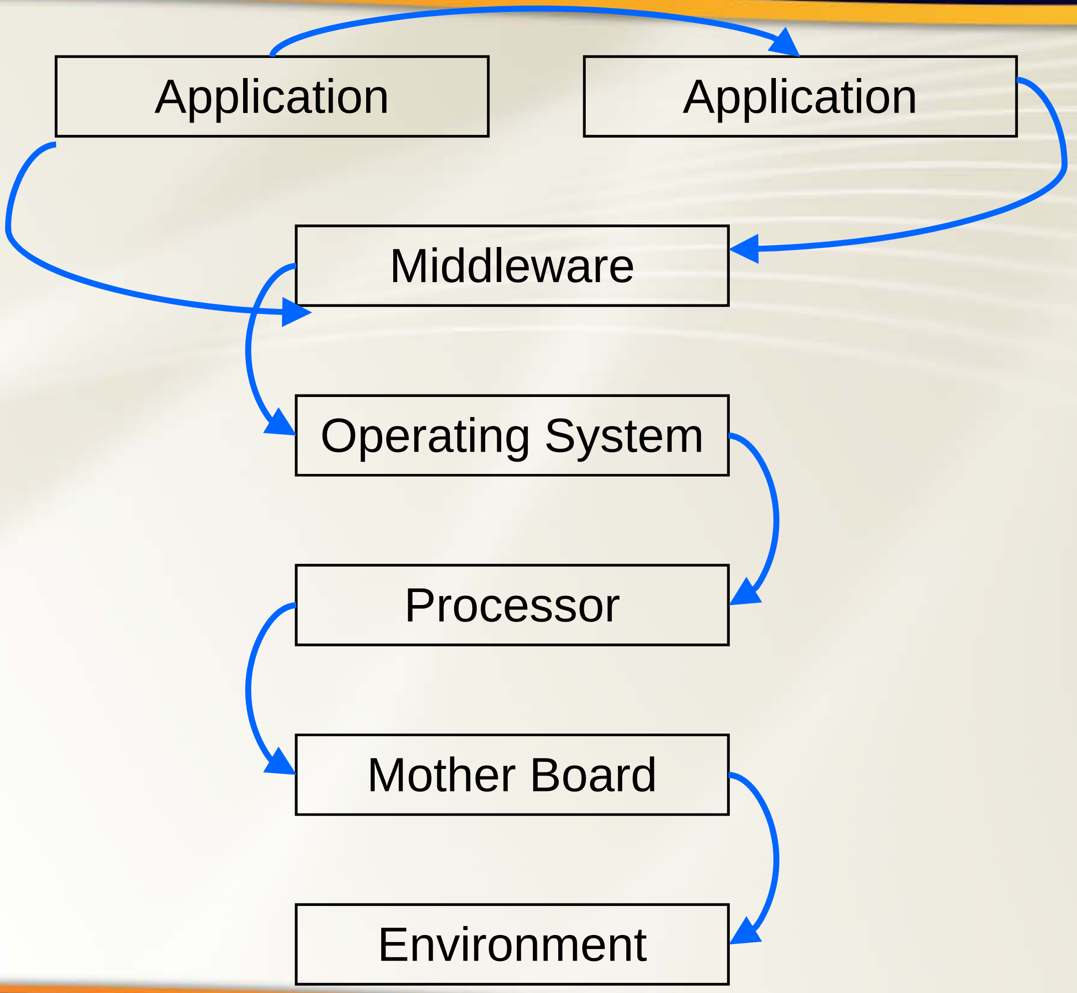
Mother Board

Environment

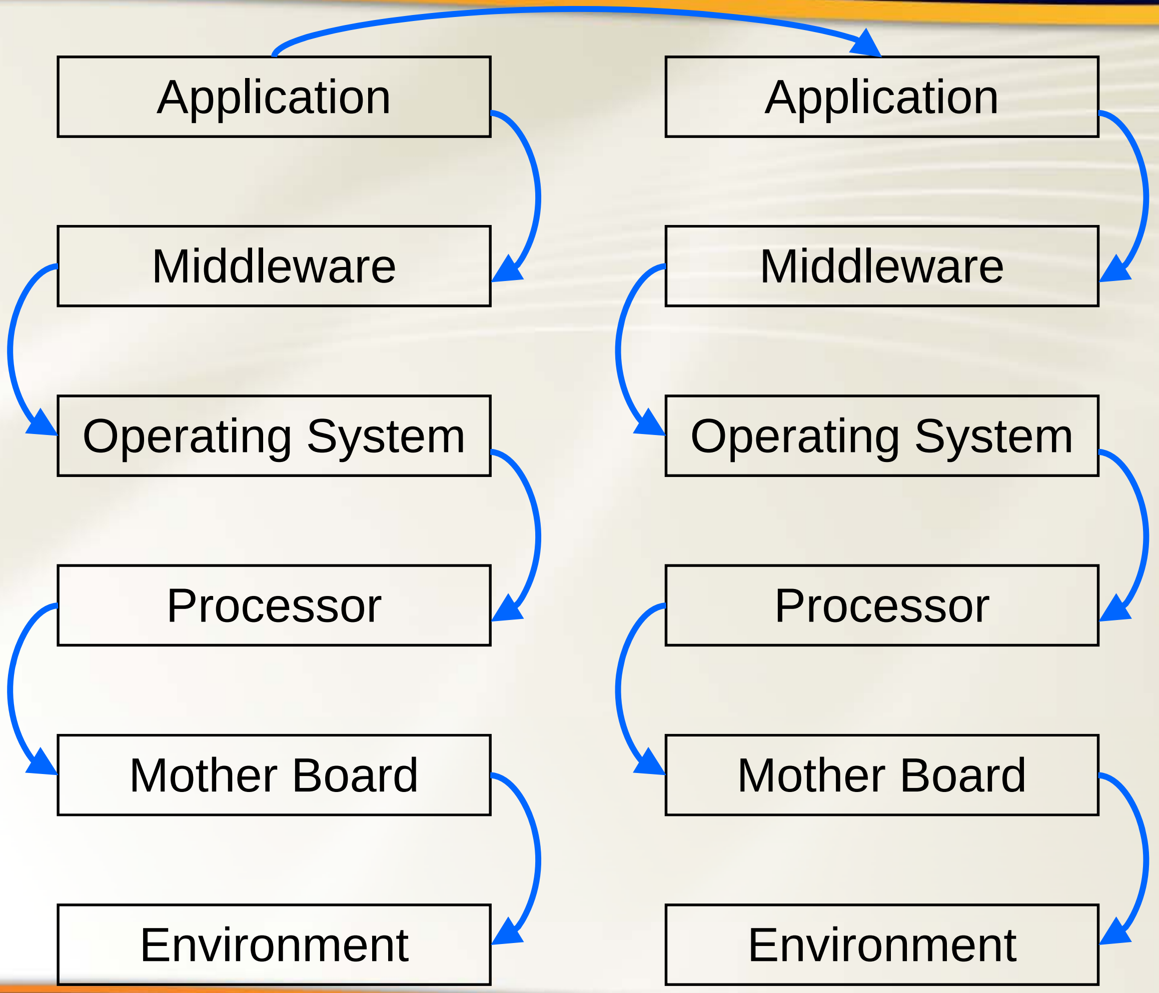
Trust Distribution

- The *real* complication is trust distributed among components and among layers

Intra-Node Distributed Trust



Inter-Node Distributed Trust



Layered Assurance Assumptions

- Each layer depends upon the properties of the layer beneath it
- The purpose of assurance at any layer is to enable assurance of the layers above it
- Higher layers don’t violate properties of lower layers
- Lower layers are independent of and benign to upper layers

Separation Enables Layered Assurance

- Separation among modules and layers
  - Elimination of side effects
  - Controlled information flow

Separation and Information Flow Control

- An intuitive boxes and arrows diagram
- Boxes encapsulate data
  - Access only local state and incoming communications
  - i.e., they are state machines
- Arrows are channels for information flow
  - Strictly unidirectional
  - Absence of arrows is often crucial
- Flaws in design or implementation might blur the separation among components
- Flaws in design or implementation might add unintended communication paths

Policy Enforcement Architecture

- Some boxes are trusted to enforce local security policies
- Trusted boxes are to be as simple as necessary
  - i.e., the principle of least privilege
  - The only practical way to achieve high assurance
- Decompose policy architecture into boxes and arrows to achieve this
- For now, assume those boxes and arrows are free

Policy Enforcement Assurance

- Construct assurance that each trusted component enforces its local policy
- Then provide an argument that the local policies combine to achieve the overall system policy
- This is done formally for critical components
  - Verified by mathematical proof

What Layered Assurance Needs

- Language that defines what it means for a component to satisfy a policy under assumptions about its environment
- Tool that verifies the policy of one component supports the assumptions of another
- An infrastructure that provides trustworthy data separation and controlled information flow

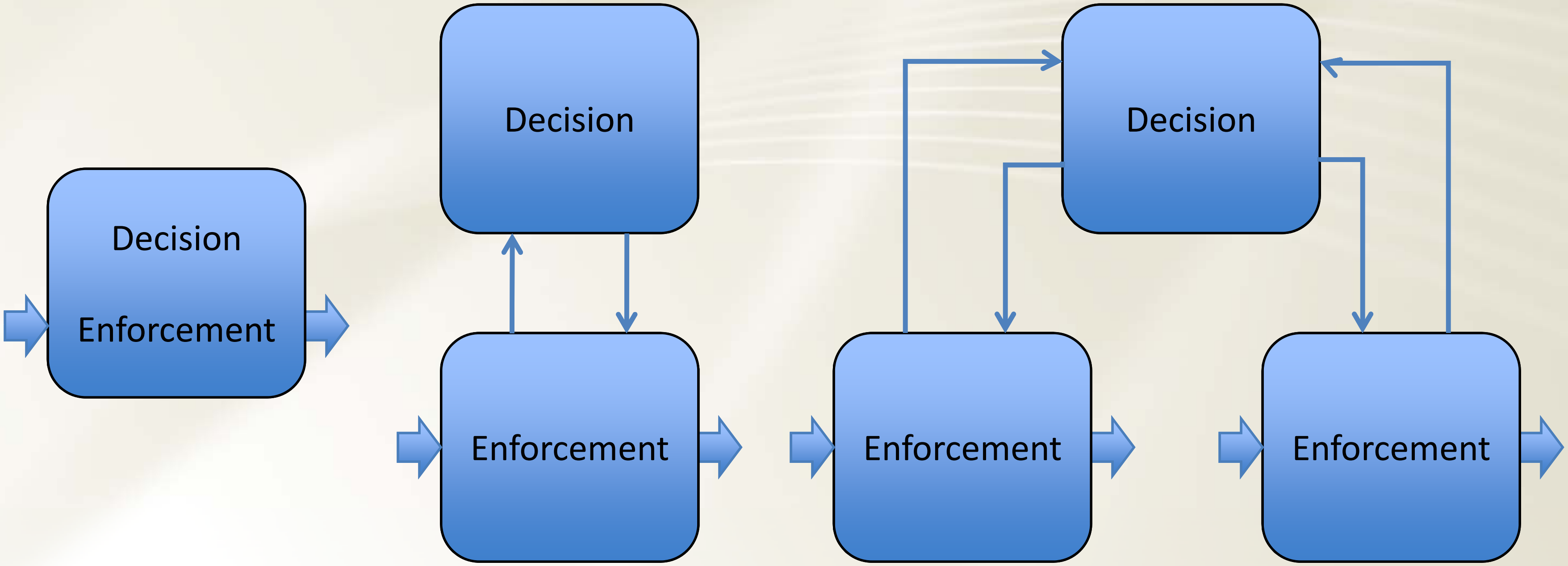
Layered Assurance Ecosystem

- Layered Assurance Workshops
  - 2007 and 2008 in Baltimore, MD
    - proceedings available
  - 2009 workshop in planning
- Research Projects
  - AFRL Information Directorate sponsored research
  - Rushby, DeLong, Boettcher, etc.

Distribution of Trust in SOA

- Still have "boxes and arrows" diagrams
- All previously discussed principles apply
- Some boxes implement the system security policy
  - Policy decision points
  - Policy enforcement points
  - Location transparency
  - 1:1 or N:1 relationships

Policy Decision and Enforcement



SOA Information Flow

- Communications between decision point and enforcement point must be secure
  - Confidentiality
  - Integrity
  - Availability
- Enforcement point communications
  - Non-bypassable with high assurance
  - Infrastructure must protect the enforcement point

Protection of Enforcement Point

- Strong identity verification
  - Nodes within distributed systems
  - Applications within authenticated nodes
- Authorization of information flow to/from enforcement point
  - ***Not*** authorization of the user!
- Secure configuration of all distributed nodes in enclave
  - Consistency of policy data
- Bandwidth provisioning and partitioning
  - Network resources: bandwidth, resources, buffers, etc.
  - Suppression of covert channels

Awarding Industry for Layered Assurance

- Incorporating considerations for portability, maintainability, technology insertion, vendor independence, and reusability
- Implementing a layered and modular system
- Eliminating inter-component dependencies
- Collaborating with the Government and within industries
- Reducing development cycle time
- Using open, standards based interfaces
- Enabling rapid technology insertion

High assurance of any kind can’t happen until there is an incentive for both industry and government to support it